Claims:

What is claimed is:

1. A framework architecture system for allowing a client application to communicate with a server component application, comprising:

a server engine for providing client access to the server, said server engine further including:

a server component for providing a service;

an implementation within said server component for providing functions of said service; and,

an interface mechanism for allowing a client application to access said implementation

2. The framework architecture system of claim 1 wherein said server component includes a plurality of implementations, and wherein said interface mechanism allows said client application to select and access one of said plugin implementations.

3. The framework architecture system of claim 1 wherein said implementation is a plugin implementation and can be replaced by another implementation at run time.

4. The framework architecture system of claim 1 wherein said interface provides an interface definition specification for the functions provided by said implementation of said server component.

5. The framework architecture system of claim 4 wherein said interface presents a data type structure according to said interface definition of which each

member of said data type structure is a function pointer to the implementation functions implementing the services defined by said interface.

6. The framework architecture system of claim 1 further comprising:

an interface namespace containing a record for each interface together with an interface identifier.

7. The framework architecture system of claim 1 wherein each interface is associated with a version of said interface.

8. The framework architecture system of claim 1 wherein there exists multiple implementations for said interface.

9. The framework architecture system of claim 1 wherein there exists multiple interfaces providing the same implementation for said server component.

10. The framework architecture system of claim 1 wherein said implementation inherits from another implementation of the same interface a subset of interface methods and functions.

11. The framework architecture system of claim 3 wherein implementations are plugged into said engine and removed from said engine by a registration process.

12. The framework architecture system of claim 1 wherein said implementation is stored in a container that is loaded by the engine at run-time

13. The framework architecture system of claim 12 wherein said container is a dynamic loadable library.

14. The framework architecture system of claim 12 wherein said container contains multiple plugins.

15. The framework architecture system of claim 1 wherein said implementation may include an interceptor for adding services to said server component.

16. The framework architecture system of claim 15 wherein said interceptor is a fanout interceptor which during the instantiation of an interface implementation causes a plurality of intercepting plugins as specified by an interception sequence attribute to be also instantiated, and wherein subsequent method invocations by the client results in invocation of the corresponding methods of intercepting plugins in the order specified.

17. The framework architecture system of claim 15 wherein said interceptor is a stack interceptor which during the realization of an interface implementation causes each plugin in an interception sequence to be instantiated in turn.

18. The framework architecture system of claim 1 further including a registry for persistent storage of implementations.

19. The framework architecture system of claim 1 further including a realization mechanism for allowing a client application to realize an implementation, wherein said implementation includes a v table, private data store, and per-instance data structure.

20. The framework architecture system of claim 19 wherein the realization mechanism includes logic for retrieving information stored within the plugins v table, private data store, and per-instance data structure, copying said information to a proxy v table, and returning a pointer to the proxy v table to the client.

21. The framework architecture system of claim 20 wherein the client uses this pointer to thereafter communicate with the interface implementation.

22. The framework architecture system of claim 1 wherein said implementation is a software personality.

23. The framework architecture system of claim 22 wherein said software personality is one of Tuxedo, Jolt, or AMS.

24. The framework architecture system of claim 23 wherein said personality includes a programming attitude.

25. The framework architecture system of claim 24 wherein said programming attitude is one os C, OLE, or Cobol.

26. The framework architecture system of claim 1 wherein said implementation is a software extension.

27. A method of allowing a client application to communicate with a server application via a framework architecture, comprising the steps of:

providing a server engine for providing client access to the server, said server engine further including:

a server component for providing a service;

an implementation within said server component for providing functions of said service; and,

an interface mechanism for allowing a client application to access said implementation.

28. The method of claim 27 further including:

allowing said client application to select and access one of a plurality of plugin implementations provided by said interface.

29. The method of claim 27 wherein said implementation is a plugin implementation and can be replaced by another implementation at run time.

30. The method of claim 27 wherein said interface provides an interface definition specification for the functions provided by said implementation of said server component.

31. The method of claim 30 wherein said interface presents a data type structure according to said interface definition of which each member of said data type structure is a function pointer to the implementation functions implementing the services defined by said interface.

32. The method of claim 27 further comprising:

maintaining an interface namespace containing a record for each interface together with an interface identifier.

33. The method of claim 27 further including:

associating each interface with a version of said interface.

34. The method of claim 27 wherein there exists multiple implementations for said interface.

35. The method of claim 27 wherein there exists multiple interfaces providing the same implementation for said server component.

36. The method of claim 27 wherein said implementation inherits from another implementation of the same interface a subset of interface methods and functions.

37. The method of claim 29 further including:

registering an implementation as a plugin in said engine via a registration process.

38. The method of claim 27 wherein said implementation is stored in a container that is loaded by the engine at run-time.

39. The method of claim 38 wherein said container is a dynamic loadable library.

40. The method of claim 38 wherein said container contains multiple plugins.

41. The method of claim 27 further including:

combining said implementation function with an interceptor that adds services to said server component.

42. The method of claim 41 wherein said interceptor is a fanout interceptor which during the instantiation of an interface implementation includes the step of:

causes a plurality of intercepting plugins as specified by an interception sequence attribute to be also instantiated, and wherein subsequent method invocations by the client results in invocation of the corresponding methods of intercepting plugins in the order specified.

43. The method of claim 41 wherein said interceptor is a stack interceptor which during the realization of an interface implementation includes the step of:

causes each plugin in an interception sequence to be instantiated in turn.

44. The method of claim 27 further including storing implementations within a registry.

45. The method of claim 27 further including:

providing a realization mechanism for allowing a client application to realize an implementation, wherein said implementation includes a v table, private data store, and per-instance data structure.

46. The method of claim 45 further including:

retrieving information stored within the plugins v table, private data store, and per-instance data structure, copying said information to a proxy v table, and returning a pointer to the proxy v table to the client.

47. The method of claim 46 further including:

using said pointer to thereafter communicate with the interface implementation.

48. The method of claim 27 wherein said implementation is a software personality.

49. The method of claim 48 wherein said software personality is one of Tuxedo, Jolt, or AMS.

50. The method of claim 49 wherein said personality includes a programming attitude.

51. The method of claim 50 wherein said programming attitude is one os C, OLE, or Cobol.

52. The method of claim 27 wherein said implementation is a software extension.